Exercise: Packing and Paging

1 Missing cow

Consider you are a cow getting lost on a misty grassland. Nearby there is a very long fence. You know that somewhere at the fence, there is a hole so you can escape. However, you don't know where the hole is nor which direction the hole is in. Furthermore, because it is so foggy, you see the hole only when it is exactly in front of you. In such a situation, all you can do is walk straightly toward one direction and search for the hole, or turn around, walk toward the other direction, and search for the hole. You want to escape using as small a total walking distance as possible.

Answer the following questions:

1. Let your starting position be the origin and the fence is the x-axis. Assume that the hole is at position n (where n can be positive or negative), what is the optimal solution cost?

2. Consider the following algorithm ALG$_1$ that first go to position 1, then go to position -1, and then 2, -2, 4, -4, ... (Figure 1):

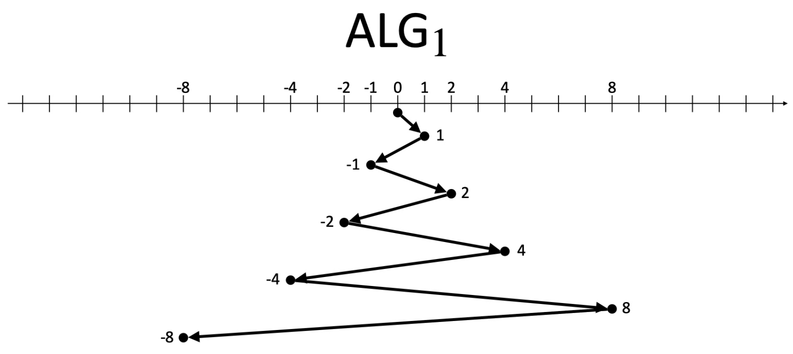


Figure 1: Cow path for ALG$_1$

   (a) Find an adversary for ALG$_1$ on the *right* hand side of the cow and give a lower bound of ALG$_1$'s competitive ratio.

   (b) Find an adversary for ALG$_1$ on the *left* hand side of the cow and give a lower bound of ALG$_1$'s competitive ratio.

   (c) From (a) and (b), which adversary gives a stronger lower bound?

   (d) Prove that ALG$_1$ is 13-competitive.

3. Consider the following algorithm ALG_2: First go to 1, then go to -2, 4, -8, 16, $\cdots$ (Figure 2.

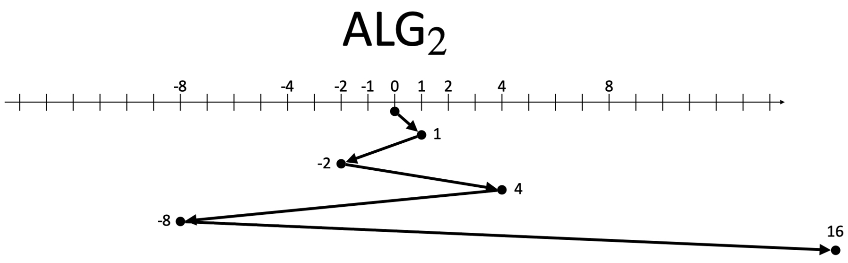


Figure 2: Cow path for ALG_1

   (a) By intuition, do you think this ALG_2 is better than ALG_1 or worse?

   (b) Show that this algorithm is 9-competitive.

2 Applying doubling technique

Recall the ski rental problem with the buying price of B and the renting price of 1 per day. Use the doubling technique to analyze the optimal online algorithm that buys a pair of skis on the B-th day.

(i) Let the parameter r be B and adapt the analysis for doubling.

(ii) Let the parameter r be 2 and adapt the analysis for doubling. What happens if $B = 2^k$ for some integer k, and what happens if $B \neq 2^k$?

3 NextFit algorithm

Recall that we introduced the FirstFit algorithm for the BinPacking problem during the lecture. Now consider another algorithm **NextFit**:

`NextFit` Algorithm

When a new item arrives, put it into the *last* bin if the item fits into that bin.
Otherwise, close the last bin, open a new bin, and put this new item to this new bin.

1. Claim that in the NextFit solution, the total size of jobs in any two consecutive bins is at least 1.

2. Prove that NextFit $\leq 2 \cdot \mathrm{OPT}$.

4 Another lower bound of First-Fit

Recall the adversary for First-Fit algorithm introduced in the lecture that contains m items of size $\frac{1}{6} - 2\varepsilon$, m items of size $\frac{1}{3} + \varepsilon$, and m items of size $\frac{1}{2} + \varepsilon$. Note that the largest denominator 6 in this adversary is a *perfect number*. That is, it is equal to the sum of its positive divisors, excluding the number itself. (Ex: 6's divisors are $1, 2, 3, 6$, and $6 = 1 + 2 + 3$.) The fact of 6 being a perfect number gives us the convenience where $\frac{1}{6} + \frac{1}{3} + \frac{1}{2} = \frac{1+2+3}{6} = \frac{6}{6} = 1$, and the adversary is designed based on that.

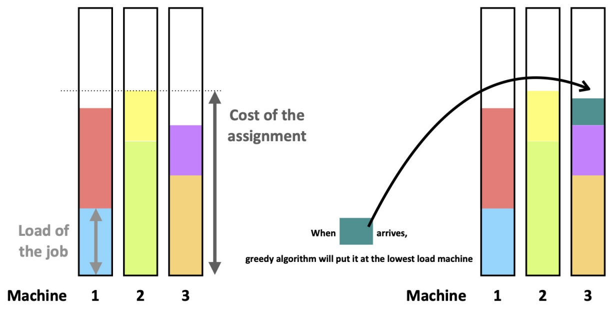
Consider the next perfect number, 28, whose divisors are $1, 2, 4, 7, 14, 28$, and $1+2+4+7+14 = 28$. What is the best adversary against First-Fit you can come up with? Is it a stronger lower bound than the one we saw in the lecture? Can you see why?

5 Online load balancing

There is a finite set of m machines. A sequence of n jobs is arriving where each job J_i is specified by its processing load ℓ_i. Each job must be assigned to exactly one of the machines upon arrival. The total load of a machine is the sum of loads of the jobs assigned to it. The cost of an assignment is the maximum total load among the machines. The goal of the LoadBalancing problem is to find an assignment with the minimum cost.

Consider the greedy algorithm (also see the illustration):

`Assign each arriving job` r_i `to the machine with the lowest load (breaking ties arbitrarily).`



1. Consider any job J_k. Claim that the cost of the optimal assignment is at least ℓ_k.

2. Claim that the cost of the optimal assignment is at least $\frac{\sum_{j=1}^{n} \ell_j}{m}$

   (Hint: Use the similar argument for bin packing optimal cost lower bound.)

3. Show that the greedy algorithm is $2 - \frac{1}{m}$-competitive.

6 Last-In-First-Out

Consider the algorithm `LIFO` (Last-In-First-Out) for the Paging problem:

`LIFO` (Last-In-First-Out) Algorithm

When there is a page fault and the cache is full, replace the page that has been copied into the cache the most recently.

Answer the following questions:

1. Given a sequence of n page requests, show that `LIFO` is at most $O(\frac{n}{k})$-competitive.

2. Show that the analysis in (a) is tight.